Read a shapefile, and export to wkt and persist in Azure blob storage

This is using Apache Sedona spatial code.

Required libraries:

- org.apache.sedona:sedona-core-3.0_2.12:1.0.1-incubating
- org.apache.sedona:sedona-sql-3.0_2.12:1.0.1-incubating
- ~~org.apache.sedona:sedona-viz-3.0_2.12:1.0.1-incubating~~
- org.apache.sedona:sedona-python-adapter-3.0_2.12:1.0.1-incubating
- org.datasyslab:geotools-wrapper:geotools-24.1

Spark Config:

- ~~spark.kryo.registrator org.apache.sedona.viz.core.Serde.SedonaVizKryoRegistrator~~
- spark.serializer org.apache.spark.serializer.KryoSerializer

```
import org.apache.spark.serializer.KryoSerializer
import org.apache.sedona.core.serde.SedonaKryoRegistrator
import org.apache.sedona.core.formatMapper.shapefileParser.ShapefileReader;
import org.apache.sedona.core.spatialRDD.SpatialRDD;
import org.apache.sedona.sql.utils.SedonaSQLRegistrator;
SedonaSQLRegistrator.registerAll(spark)
```

```
import org.apache.spark.serializer.KryoSerializer
import org.apache.sedona.core.serde.SedonaKryoRegistrator
import org.apache.sedona.core.formatMapper.shapefileParser.ShapefileReader
import org.apache.sedona.core.spatialRDD.SpatialRDD
import org.apache.sedona.sql.utils.SedonaSQLRegistrator
```

```
%python
def createBlobMountPoint(storageAccountKey, container, scope, key,
rootMountPoint="/mnt"):
  """
  Create a dbfs mount point for a container in Azure Blob storage. (key) will
determine permissions.
  Args:
    storageAccountKey: Azure key vault key name for the storage account
    rootMountPoint: root folder for the mount point, typically '/mnt'
    container: Azure blob container to be read or written
    scope: Databricks security scope
    key: name of key in Azure key vault
  Date:
    2021-07-26
  Version:
    1.0
  """
  # create r/? mount point
  # storage account name: iamosley8dasource
  # the r/o secrets key name: da-source-read-only
  # the storage account key name: da-source-name
  # scope: ReadOnlyDaSourceBlob
  # container:

  print(dbutils.secrets.listScopes())
  # then use it to list the key names
  print(dbutils.secrets.list(scope))
  print(dbutils.secrets.get(scope=scope, key=key))

  # create a mount point for the files

  mountPoint = rootMountPoint + "/" + container

  if mountPoint in [mnt.mountPoint for mnt in dbutils.fs.mounts()]:
    dbutils.fs.unmount(mountPoint)

  # Add the Storage Account, Container, and reference the secret to pass the
SAS Token
  STORAGE_ACCOUNT = dbutils.secrets.get(scope=scope, key=storageAccountKey)
  CONTAINER = container
  SASTOKEN = dbutils.secrets.get(scope=scope, key=key)

  # build credentials
  SOURCE =
"wasbs://{container}@{storage_acct}.blob.core.windows.net/".format(container=CO
NTAINER, storage_acct=STORAGE_ACCOUNT)
```

```
  URI = "fs.azure.sas.{container}.
{storage_acct}.blob.core.windows.net".format(container=CONTAINER,
storage_acct=STORAGE_ACCOUNT)

  try:
    dbutils.fs.mount(
      source=SOURCE,
      mount_point=mountPoint,
      extra_configs={URI:SASTOKEN})
  except Exception as e:
    if "Directory already mounted" in str(e):
      pass # Ignore error if already mounted.
    else:
      raise e

  print(mountPoint)
  print(dbutils.fs.ls(mountPoint))
```

```
%python
# create the mount point for reading the DA shapefile
createBlobMountPoint(storageAccountKey="da-source-name", container="census-
2016-source", scope="ReadOnlyDaSourceBlob", key="da-source-read-only")
```

```
[SecretScope(name='ReadOnlyDaSourceBlob')]
[SecretMetadata(key='da-source-name'), SecretMetadata(key='da-source-read-onl
y'), SecretMetadata(key='da-source-write')]
[REDACTED]
/mnt/census-2016-source has been unmounted.
/mnt/census-2016-source
[FileInfo(path='dbfs:/mnt/census-2016-source/98-401-X2016044_English_CSV_data.
csv', name='98-401-X2016044_English_CSV_data.csv', size=14570396610), FileInfo
(path='dbfs:/mnt/census-2016-source/98-401-X2016044_English_meta_crlf.txt', na
me='98-401-X2016044_English_meta_crlf.txt', size=226465), FileInfo(path='dbf
s:/mnt/census-2016-source/lda_000b16a_e.dbf', name='lda_000b16a_e.dbf', size=3
8311491), FileInfo(path='dbfs:/mnt/census-2016-source/lda_000b16a_e.prj', name
='lda_000b16a_e.prj', size=604), FileInfo(path='dbfs:/mnt/census-2016-source/l
da_000b16a_e.shp', name='lda_000b16a_e.shp', size=190048652), FileInfo(path='d
bfs:/mnt/census-2016-source/lda_000b16a_e.shx', name='lda_000b16a_e.shx', size
=452812)]
```

```
// convert the Shapefile into an RDD
val MOUNTPOINT = "/mnt/census-2016-source"
val spatialRdd = ShapefileReader.readToPolygonRDD(sc, MOUNTPOINT +
"/lda_000b16a_e.*")
```

```
MOUNTPOINT: String = /mnt/census-2016-source
spatialRdd: org.apache.sedona.core.spatialRDD.PolygonRDD = org.apache.sedona.c
ore.spatialRDD.PolygonRDD@ec67df
```

```
%python
# create a mount point to write out the Shapefile converted to WKT
createBlobMountPoint(storageAccountKey="da-source-name", container="census-
2016-silver", scope="ReadOnlyDaSourceBlob", key="da-source-write")
```

```
[SecretScope(name='ReadOnlyDaSourceBlob')]
[SecretMetadata(key='da-source-name'), SecretMetadata(key='da-source-read-onl
y'), SecretMetadata(key='da-source-write')]
[REDACTED]
/mnt/census-2016-silver has been unmounted.
/mnt/census-2016-silver
[]
```

```
// write the WKT
// this method will *NOT* overwrite any existing files
var MOUNTPOINT = "/mnt/census-2016-silver"
spatialRdd.saveAsWKT(MOUNTPOINT + "/DA_etc.wkt")
```

```
MOUNTPOINT: String = /mnt/census-2016-silver
```

	path	name	size
1	dbfs:/mnt/census-2016-silver/DA_etc.wkt/_SUCCESS	_SUCCESS	0
2	dbfs:/mnt/census-2016-silver/DA_etc.wkt/part-00000	part-00000	364721239

Showing all 2 rows.